Docket No. AUS920030550US1 *PATENT*

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re application of: **DeWitt, Jr. et al.**	§	
	§	Group Art Unit: **2181**
Serial No. **10/757,237**	§	
	§	Examiner: **Lai, Vincent**
Filed: **January 14, 2004**	§	
	§	
For: **Autonomic Method and Apparatus for Counting Branch Instructions to Improve Branch Predictions**	§	

Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

35525
PATENT TRADEMARK OFFICE
CUSTOMER NUMBER

APPEAL BRIEF (37 C.F.R. 41.37)

This brief is in furtherance of the Notice of Appeal, filed in this case on December 4, 2006.

A fee of $500.00 is required for filing an Appeal Brief. Please charge this fee to IBM Corporation Deposit Account No. 09-0447. No additional fees are believed to be necessary. If, however, any additional fees are required, I authorize the Commissioner to charge these fees which may be required to IBM Corporation Deposit Account No. 09-0447. No extension of time is believed to be necessary. If, however, an extension of time is required, the extension is requested, and I authorize the Commissioner to charge any fees for this extension to IBM Corporation Deposit Account No. 09-0447.

REAL PARTY IN INTEREST

The real party in interest in this appeal is the following party: International Business Machines Corporation of Armonk, New York.

RELATED APPEALS AND INTERFERENCES

With respect to other appeals or interferences that will directly affect, or be directly affected by, or have a bearing on the Board's decision in the pending appeal, there are no such appeals or interferences.

STATUS OF CLAIMS

A. TOTAL NUMBER OF CLAIMS IN APPLICATION

Claims in the application are: 1-21.

B. STATUS OF ALL THE CLAIMS IN APPLICATION

1. Claims canceled: NONE.
2. Claims withdrawn from consideration but not canceled: NONE.
3. Claims pending: 1-21.
4. Claims allowed: NONE.
5. Claims rejected: 1-21.
6. Claims objected to: 6, 9, 13, and 20.

C. CLAIMS ON APPEAL

The claims on appeal are: 1-21.

Note: Claims 6, 9, 13, and 20 are objected to as containing a grammatical error. If the claims under appeal are sustained, claims 6, 9, 13, and 20 will be amended for proper grammar.

STATUS OF AMENDMENTS

There are no amendments after the final rejection.

SUMMARY OF CLAIMED SUBJECT MATTER

Independent claim 1:

The present invention provides a method of performing branch prediction in a computer program. (Specification, page 24, lines 27-30) The present invention identifies a plurality of branch instructions for application code being compiled. (Specification, page 25, lines 1-5) The present invention associates a plurality of hardware counters with the plurality of branch instructions. (Specification, page 18, lines 4-23) The present invention uses the plurality of hardware counters to autonomically count all of the plurality of branch instructions that are executed in parallel to generate a plurality of branch statistics. (Specification, page 25, lines 5-30) The present invention predicts branches to be taken using the plurality of branch statistics to form branch predictions. (Specification, page 26, lines 7-9) The present invention prefetches the plurality of branch instructions using the plurality of branch predictions. (Specification, page 26, lines 9-24)

Independent claim 8:

The present invention provides for a branch prediction apparatus. (Specification, page 19, lines 7-11) The present invention provides a compiler that identifies a plurality of branch instructions for application code being compiled. (Specification, page 19, lines 19-21) The present invention provides a plurality of hardware counters associated with the plurality of branch instructions of the application code. (Specification, page 20, line 27, to page 21, line 1) The present invention provides a plurality of branch statistic fields for storing a plurality of branch statistics associated with the plurality of branch instructions. (Specification, page 19, lines 11-13) The present invention provides wherein when a branch instruction in the plurality of branch instructions is executed in the application code, a hardware counter of the plurality of hardware counters autonomically counts all of the plurality of branch instructions that are executed and updates in parallel branch statistics in the plurality of branch statistic fields. (Specification, page 25, lines 5-30) The present invention provides a processor that predicts branches to be taken

using the plurality of branch statistics to form branch predictions. (Specification, page 19, line 28, to page 20, line 2 and page 26, lines 7-9) The present invention provides a processor that prefetches the plurality of branch instructions using the branch predictions. (Specification, page 19, line 28, to page 20, line 2 and page 26, lines 9-24)

The apparatus recited in claim 8, as well as dependent claims 9-14, may be an apparatus comprised of compiler **322**, branch statistic fields **302-308** and processor **316** of **Figure 3** and hardware counters **241** and **242** of **Figure 2**. The compiler, counters, branch statistic fields, and processor perform the steps described in the specification at page 24, line 27, to page 26, line 24, without undue experimentation.

Independent claim 15:

The present invention provides for a computer program product in a recordable-type computer readable medium. (Specification, page 27, line 14, to page 28, line 2) The present invention provides instructions for identifying a plurality of branch instructions for application code being compiled. (Specification, page 24, lines 1-5) The present invention provides instructions for associating a plurality of hardware counters with the plurality of branch instructions. (Specification, page 18, lines 4-23) The present invention provides instructions for autonomically counting all of the plurality of branch instructions that are executed in parallel using the plurality of hardware counters to thereby generate a plurality of branch statistics. (Specification, page 25, lines 5-30) The present invention provides instructions for predicting branches to be taken using the plurality of branch statistics to form branch predictions. (Specification, page 26, lines 7-9) The present invention provides instructions for executing the application code using the branch predictions. (Specification, page 26, lines 9-24)

A person having ordinary skill in the art would be able to derive computer instructions on a computer readable medium as recited in claim 15, as well as dependent claims 16-21, given **Figures 5** and **6,** and the corresponding description in the specification at page 24, line 27, to page 26, line 24, without undue experimentation.

GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL

The grounds of rejection to review on appeal are as follows:

A. GROUND OF REJECTION (Claims 1-21)

Whether claims 1-21 are obvious under 35 U.S.C. § 103(a) over Holmberg (U.S. Patent No. 6,233,679 B1).

ARGUMENT

A. 35 U.S.C. § 103, Obviousness, Claims 1-21

Claim 1 is representative of the claims in this group and reads as follows:

1. A method of performing branch prediction in a computer program, comprising the steps of:

identifying a plurality of branch instructions for application code being compiled;

associating a plurality of hardware counters with the plurality of branch instructions;

using the plurality of hardware counters to autonomically count all of the plurality of branch instructions that are executed in parallel to generate a plurality of branch statistics;

predicting branches to be taken using the plurality of branch statistics to form branch predictions; and

prefetching the plurality of branch instructions using the plurality of branch predictions. (emphasis added)

Appellants respectfully submit that Holmberg does not teach or suggest identifying a plurality of branch instructions for application code being compiled and using the plurality of hardware counters to autonomically count all of the plurality of branch instructions that are executed in parallel to generate branch statistics.

As to claim 1, the Examiner states:

As per claim 1, Holmberg teaches a method of performing branch prediction (See column 3, lines 35-37: Branch prediction is the main purpose of the Holmberg invention) in a computer program, comprising the steps of:

identifying a plurality of branch instructions for application code being complied (See column 3, lines 50-52: Processor identifies type of instruction and thus will know which are branch instructions);

associating a plurality of hardware counters with the plurality of branch instructions (See column 4, lines 56-63: Multiple counters are used-each for counting various actions);

using the plurality hardware counter to autonomically count branch instructions that are executed in parallel to generate a plurality of branch statistics (See column 4, lines 64-65: Counters are used from providing statistics);

predicting branches to be taken using the plurality of branch statistics (See column 4, lines 64-65: Counter statistics are used to set branch prediction bits) to form branch predictions; and

> prefetching the plurality of branch instructions using the plurality of branch predictions (See column 4, lines 1-3).
>
> Holmberg does not teach counting all of the plurality of branch instructions that are executed in parallel.
>
> It would have been obvious to a person having ordinary skill in the art at the time the invention was made to have modified Holmberg such that counting all of the plurality of branch instructions that are executed in parallel would be done. Holmberg teaches that statistics should be collected for all conditional branches (See column 2, lines 60-64) and that braches are to be run in parallel (See column 2, lines 50-53). This would necessitate the need/desire to count all branch instructions in parallel. Holmberg goes on to teach the use of a plurality of hardware counters for collecting statistics for the plurality of branch instructions (See column 2, lines 66-67), thus suggesting that it there is a desire to count all of the plurality of branch instructions that are executed in parallel.

Office Action dated October 5, 2006, pages 4-5.

Thus, the Examiner clearly acknowledges that "Holmberg does not teach counting all of the plurality of branch instructions that are executed in parallel."

Holmberg is directed to a branch prediction system that uses a scanning mechanism for scanning the program memory for conditional branch instructions during the running of the program. When finding such an instruction, the system records during a preset time interval, the statistics for that specific conditional branch instruction and sets a branch prediction bit in the instruction accordingly. The system then starts to scan for the next conditional branch instruction in the program memory. (see Holmberg, Abstract)

Thus, Holmberg only records statistics for one specific conditional branch instruction at a time. In contradistinction, the present invention associates a plurality of hardware counters with a plurality of branch instructions and uses the plurality of hardware counters to autonomically count all of the plurality of branch instructions that are executed in parallel to generate branch statistics. As a result, all of the branch instructions are counted as they are executed as opposed to Holmberg who states:

> Thus, the background program begins with scanning or searching the program memory for the first conditional jump instruction in a block 303. When finding the first conditional branch instruction the corresponding program memory address is loaded into the Measured Address Register (MAR) in a block 305. Thereupon the program checks all counters used for collecting the statistics in a block 307.
>
> Next, all counters are started in a block 309. The background program now waits for statistics to be collected. The counters are incremented each time

> the program from which statistics are collected executes the conditional branch instruction associated with the address stored in the MAR and when the corresponding branch is taken, respectively, if the implementation as described in conjunction with FIG. 2 is used. The statistics for a specific conditional branch instruction are collected for a predefined time as indicated in block 311, which can be equally long for each conditional branch instruction.

(Holmberg, column 5, lines 16-25)

In this section, Holmberg specifically describes the system scans for the first conditional jump instruction. Once the first conditional jump instruction is encountered, it is loaded into a measured address register. Then, Holmberg increments the counter each time the program from which statistics are collected executes the conditional branch instruction associated with the address stored in the MAR and when the corresponding branch is taken. Thus, Holmberg only records statistics for one conditional jump instruction and then moves onto the next conditional branch instruction. The present invention counts all of the plurality of branch instructions that are executed in parallel to generate branch statistics.

The Examiner further states "It would have been obvious to a person having ordinary skill in the art at the time the invention was made to have modified Holmberg such that counting all of the plurality of branch instructions that are executed in parallel would be done." The Examiner alleges that support modifying Holmberg is provided in the following sections:

> The method makes it possible to predict multiple conditional branch instructions in parallel, which for example can be needed in superscalar processors for getting good branch prediction accuracy.
>
> The program performance depends on the execution statistics for the predicted branch only, not on interaction with other conditional branch instructions in other programs.
>
> However, there are some conditions that must be met for the semi-static branch prediction mechanism to work well, i.e. to provide a good branch prediction.
>
> i) The sample program must be executed for a relatively long time and have approximately the same behaviour during that time since it takes some time for the background program to scan the program for all conditional branch instructions, and collect reliable statistics.
>
> ii) It must be possible to include the branch prediction bit.
>
> iii) It must be possible to include a hardware counter or counters for collecting execution statistics.

(Holmberg, column 2, lines 50-67)

In this section, Holmberg describes predicting multiple conditional branches in parallel, not counting multiple conditional branches in parallel. As stated previously, Holmberg specifically describes the system scans for the first conditional jump instruction. Once the first conditional jump instruction is encountered, it is loaded into a measured address register. Then, Holmberg increments the counter each time the program from which statistics are collected and executes the conditional branch instruction associated with the address stored in the MAR and when the corresponding branch is taken. Thus, Holmberg only records statistics for one conditional jump instruction and then moves onto the next conditional branch instruction. Holmberg further describes that the sample program must be run for a relatively long time in order to collect reliable statistics. The reason for Holmberg running the sample program so long is because Holmberg does not teach counting all of the plurality of branch instructions that are executed in parallel, as acknowledged by the Examiner.

Additionally, Holmberg does not teach or suggest identifying a plurality of branch instructions **for application code being compiled**. As discussed above, Holmberg clearly states that the branch prediction system uses a scanning mechanism for scanning the program memory for conditional branch instructions **during the running of the program**. Thus, Holmberg records statistics for one conditional jump instruction during the running of the program and not during the compile phase.

Furthermore, Holmberg teaches away for the presently claimed invention recited in the claims by stating "The program performance depends on the execution statistics for the predicted branch only, not on interaction with other conditional branch instructions in other programs." Thus, Holmberg does not intend or need to count all of the plurality of branch instructions during a compile phase that are executed in parallel as Holmberg is only concerned with the execution statistics for the predicted branch only and not on interaction with other conditional branch instructions.

The Examiner bears the burden of establishing a *prima facie* case of obviousness based on the prior art when rejecting claims under 35 U.S.C. § 103. *In re Fritch*, 972 F.2d 1260, 23 U.S.P.Q.2d 1780 (Fed. Cir. 1992). Since the reference fails to teach or suggest identifying a plurality of branch instructions for application code being compiled and using the plurality of hardware counters to autonomically count all of the plurality of branch instructions that are

executed in parallel to generate branch statistics, the Examiner has failed to establish a *prima facie* case of obviousness, because the Examiner does not show where each and every claim limitation is taught or fairly suggested by the applied prior art.

The applied reference does not teach or suggest each and every claim limitation; therefore, Holmberg does not render claim 1 obvious. Independent claims 8 and 15 recite similar subject matter addressed above with respect to claim 1 and are allowable for similar reasons. Since claims 2-7, 9-14, and 16-21 depend from claims 1, 8, and 15, the same distinctions between Holmberg and the invention recited in claims 1, 8, and 15 apply for these claims. Additionally, claims 2-7, 9-14, and 16-21 recite other additional combinations of features not taught or suggested by the references.

Furthermore, no suggestion is present in the reference to modify the reference to include such features. That is, there is no teaching or suggestion in Holmberg that a problem exists for which identifying a plurality of branch instructions for application code being compiled and using the plurality of hardware counters to autonomically count all of the plurality of branch instructions that are executed in parallel to generate branch statistics, is a solution. To the contrary, Holmberg appears to teach recording statistics for one conditional jump instruction during the running of the program and then moves onto the next conditional branch instruction. Holmberg is not concerned with using the plurality of hardware counters to autonomically count all of the plurality of branch instructions that are executed in parallel to generate branch statistics.

One of ordinary skill in the art, being presented only with Holmberg, and without having a prior knowledge of Appellants' claimed invention, would not have found it obvious to modify Holmberg to arrive at Appellants' claimed invention, as recited in claim 1. To the contrary, even if one were somehow motivated to modify Holmberg, and it were somehow possible to modify the system, the result would not be the invention, as recited in claim 1. The resulting system would still fail to identify a plurality of branch instructions for application code being compiled and use the plurality of hardware counters to autonomically count all of the plurality of branch instructions that are executed in parallel to generate branch statistics.

In view of the above, Appellants respectfully submit that Holmberg fails to teach or suggest the features of claims 1, 8, and 15. At least by virtue of their dependency on claims 1, 8,

and 15, the features of dependent claims 2-7, 9-14, and 16-21 are not taught or suggested by Holmberg. Accordingly, Appellants respectfully request the rejection of claims 1-21 under 35 U.S.C. § 103 not be sustained.

CONCLUSION

In view of the above, Appellants respectfully submit that claims 1-21 are allowable over the cited prior art and that the application is in condition for allowance. Accordingly, Appellants respectfully request the Board of Patent Appeals and Interferences to reverse the rejections set forth in the Final Office Action.

/Francis Lammes/
Francis Lammes
Reg. No. 55,353
YEE & ASSOCIATES, P.C.
PO Box 802333
Dallas, TX 75380
(972) 385-8777

CLAIMS APPENDIX

The text of the claims involved in the appeal are:

1. A method of performing branch prediction in a computer program, comprising the steps of:

identifying a plurality of branch instructions for application code being compiled;

associating a plurality of hardware counters with the plurality of branch instructions;

using the plurality of hardware counters to autonomically count all of the plurality of branch instructions that are executed in parallel to generate a plurality of branch statistics;

predicting branches to be taken using the plurality of branch statistics to form branch predictions; and

prefetching the plurality of branch instructions using the plurality of branch predictions.

2. The method of claim 1, wherein the plurality of branch instructions are associated with the plurality of branch statistics, and wherein the plurality of branch statistics are stored in a plurality of branch statistic fields.

3. The method of claim 2, wherein the plurality of branch statistic fields store a plurality of data on an associated branch instruction, wherein a first datum of the plurality of data is accessed for branch prediction when the program is in a first mode, and wherein a second datum of the plurality of data is accessed for branch prediction when the program is in a second mode.

4. The method of claim 2, wherein the plurality of branch statistic fields include a branch count per instruction field that represents the number of times a branch is taken for that branch instruction.

5. The method of claim 1, wherein upon occurrence of a predetermined event, the computer program switches branch prediction operating modes on a conditional branch instruction.

6. The method of claim 1, wherein the plurality of branch statistics are stored in a performance instrumentation shadow cache.

7. The method of claim 1, wherein branches per instruction are counted during execution of the computer program.

8. A branch prediction apparatus, comprising:

a compiler that identifies a plurality of branch instructions for application code being compiled;

a plurality of hardware counters associated with the plurality of branch instructions of the application code;

a plurality of branch statistic fields for storing a plurality of branch statistics associated with the plurality of branch instructions;

wherein when a branch instruction in the plurality of branch instructions is executed in the application code, a hardware counter of the plurality of hardware counters autonomically

counts all of the plurality of branch instructions that are executed and updates in parallel branch statistics in the plurality of branch statistic fields;

a processor that predicts branches to be taken using the plurality of branch statistics to form branch predictions; and

the processor prefetches the plurality of branch instructions using the branch predictions.

9. The apparatus of claim 8, wherein the plurality of branch statistics are used to make branch predictions in the application code.

10. The apparatus of claim 8, further comprising a plurality of operating modes of the application code, wherein for a first branch instruction, an associated branch statistics field stores first branch statistics for a first mode of the plurality of operating modes and second branch statistics for a second mode of the plurality of operating modes.

11. The apparatus of claim 8, wherein the plurality of branch statistic fields include a branch count per instruction field that represents the number of times a branch is taken for that branch instruction.

12. The apparatus of claim 8, wherein upon occurrence of a predetermined event, the program switches branch prediction operating modes on a conditional branch instruction.

13. The apparatus of claim 8, wherein the plurality of branch statistics are stored in a performance instrumentation shadow cache.

14. The apparatus of claim 8, wherein branches per instruction are counted during execution of the program.

15. A computer program product in a recordable-type computer readable medium, comprising:

instructions for identifying a plurality of branch instructions for application code being compiled;

instructions for associating a plurality of hardware counters with the plurality of branch instructions;

instructions for autonomically counting all of the plurality of branch instructions that are executed in parallel using the plurality of hardware counters to thereby generate a plurality of branch statistics;

instructions for predicting branches to be taken using the plurality of branch statistics to form branch predictions; and

instructions for executing the application code using the branch predictions.

16. The computer program product of claim 15, wherein the plurality of branch instructions are associated with the plurality of branch statistics, and wherein the plurality of branch statistics are stored in a plurality of branch statistic fields.

17. The computer program product of claim 16, wherein the plurality of branch statistic fields store a plurality of data on an associated branch instruction, wherein a first datum of the plurality of data is accessed for branch prediction when the program is in a first mode, and

wherein a second datum of the plurality of data is accessed for branch prediction when the program is in a second mode.

18. The computer program product of claim 16, wherein the plurality of branch statistic fields include a branch count per instruction field that represents the number of times a branch is taken for that branch instruction.

19. The computer program product of claim 15, wherein upon occurrence of a predetermined event, the computer program switches branch prediction operating modes on a conditional branch instruction.

20. The computer program product of claim 15, wherein the plurality of branch statistics are stored in a performance instrumentation shadow cache.

21. The computer program product of claim 15, wherein branches per instruction are counted during execution of the computer program.

EVIDENCE APPENDIX

There is no evidence to be presented.

RELATED PROCEEDINGS APPENDIX

There are no related proceedings.